Selecting Secret Sharing Instantiations for Distributed Storage

Giulia Traverso
TU Darmstadt, Germany
gtraverso@cdc.informatik.tu-darmstadt.de

Paul Ranly
TU Darmstadt, Germany
paulmoritz.ranly@stud.tu-darmstadt.de

Denis Butin
TU Darmstadt, Germany
dbutin@cdc.informatik.tu-darmstadt.de

Johannes Buchmann
TU Darmstadt, Germany
buchmann@cdc.informatik.tu-darmstadt.de

ABSTRACT

Distributed storage systems using secret sharing enable information-theoretic confidentiality, making them especially suitable for the outsourced storage of sensitive data. In particular, proactive secret sharing enhances the confidentiality protection of such systems by periodically renewing data shares. This adds a time constraint for an attacker trying to reconstruct the initial data by collecting enough shares. Proactive secret sharing can only be carried out effectively if the participating servers (grouped in storage service providers) are reliable. The selection of participating servers is thus critical to security. In practice, data owners have little means to make an informed decision in this regard. Furthermore, optimal share allocation depends on data-owner-specific confidentiality, availability and cost requirements. Data owners also require guidance with respect to the selection of the underlying secret sharing scheme. In this paper, we introduce a novel approach to guide data owners in the instantiation of secret sharing for outsourced storage. The decision support covers both the allocation of shares to specific storage service providers, and the choice of the underlying secret sharing scheme. We realise our approach as a solver for a set of integer linear programming problems. We then dually evaluate our approach. First, we evaluate the feasability of constraint solving by implementing the linear programs in PuLP and inputting them to the GLPK linear problem solver. The evaluation involves sixty data centers from six major public cloud providers. Second, we compare the performance of hierarchical and non-hierarachical secret sharing schemes to determine if the performance loss due to the support of hierarchical structures is affordable. Ultimately, our approach aims at supporting non-expert data owners in making the most appropriate choices for the selection of a secret-sharing-based distributed storage system, based on their requirements.

CCS CONCEPTS

• **Security and privacy** → *Information-theoretic techniques*; *Distributed systems security*; *Usability in security and privacy*.

KEYWORDS

Long-term security, secure storage, secret sharing, decision support system, information-theoretic confidentiality.

ACM Reference Format:
Giulia Traverso, Paul Ranly, Denis Butin, and Johannes Buchmann. 2019. Selecting Secret Sharing Instantiations for Distributed Storage. In *Proceedings of International Workshop on Security in Cloud Computing (SCC'19).* ACM, New York, NY, USA, 11 pages.

SCC'19, July 7–12 2019, Auckland, New Zealand

1 INTRODUCTION

Sensitive digital data, such as electronic health records, require lengthy secure storage. Often, the confidentiality of such data must be protected for at least a human lifetime. A naive approach to this problem is to encrypt the data and then outsource the encrypted data to a storage server. However, this is not a viable solution to protect the confidentiality of data with stringent privacy requirements such as health records for two main reasons. First, there could always be a sudden cryptanalytic advance, where algorithms that solve the hardness problems currently used encryption schemes are based on. This enables data providers and intelligent agencies to store large amounts of encrypted data and then to decrypt them later once the hardness problems of the encryption algorithms become computationally feasible to be solved. Second, undeniable progresses have been made lately with respect to the construction of quantum computers [7, 38], which, once available, are able to break most-used public-key cryptography. In addition, availability of the outsourced data must also be guaranteed. For instance, availability and quick retrieval of electronic health records (e.g. blood group when in need of a urgent transfusion) might be a crucial to save the life of patients.

Distributed storage systems based on secret sharing [30] are relatively expensive due to the need for multiple storage servers. However, they are a promising solution to protect the confidentiality and availability of critically sensitive data sets, for which a higher cost is justifiable. Secret sharing schemes are neither vulnerable to cryptanalysic progress, nor attacks run on quantum computers because they provide *information-theoretic confidentiality*. Shares of the data to be outsourced are generated so that each share is distributed to one of the storage servers making up the distributed storage system. Information-theoretic confidentiality, also referred to as unconditional security, is provided because the outsourced data can be reconstructed only if a minimum number of shares (and, thus, storage servers) is available and no information about the data is leaked otherwise. Availability of the outsourced data is provided because secret sharing-based distributed storage systems are robust

to a certain amount of unavailable storage servers. This is again due to the fact that less shares (and, thus, less storage servers) than those generated are necessary to reconstruct the outsourced data.

Confidentiality of data is guaranteed in the long-term by *proactive secret sharing* [12], where the shares are periodically updated to cope with a mobile adversary that, over time, can compromise enough storage servers to successfully reconstruct the outsourced data. Proactive secret sharing is performed solely by the storage servers in a distributed fashion, without the intervention of the data owner. Thus, long-term confidentiality of data in secret-sharing-based distributed storage systems relies on high-performing storage servers, reliably carrying out periodically proactive secret sharing. We use the term *performance* in a broad sense. Furthermore, distributed storage systems are in practice composed of storage servers owned by different storage service providers (SSPs), so that the system is not vulnerable to a single point of failure (in this case, the key management of a SSP). This is the outsourcing scenario where, from now on, we assume data owners want to perform proactive secret sharing to protect the confidentiality of stored data.

However, despite providing confidentiality of data, the above solution where proactive secret sharing is performed across the storage servers from different SSPs does not fully address more realistic scenarios such as companies, hospitals, and institutions. The reason is twofold. On the one hand, companies, hospitals, and institutions are organized hierarchically. This means that the each member or employee covers a specific position with well defined powers and rights, even concerning the access of documents and file. These access rules should be reflected also by the secret sharing scheme used to set up the distributed storage system. On the other hand, there is no guarantee that the storage servers will reliably carry out the tasks their are supposed to perform [15] (in this case, proactive secret sharing). It is therefore extremely difficult to choose in practice high-performing storage servers to be included in the distributed storage system. This is due to both the lack of insight of data owners with respect to the quality of service of offered by SSPs in available in the marketplace and the lack of standardized and transparent means providing accurate and reliable performance figures. This issue has been also pointed out by NIST [14], making decision support an important paradigm in cloud computing. For instance, RSA [28] already announced a Trust Authority (TA) that provides decision support in the cloud. Furthermore, even with accurate performance figures in hand, data owners still lack of the cryptographic expertise to choose the right secret sharing scheme suitable for their needs and the state of the storage servers. The performance of the storage servers might indeed not be constant over time [29] and users do not often have the insight to observe this and act accordingly. Also, the secret sharing-based distributed storage system set up should protect the confidentiality of the data, while achieving the best configuration in terms of storage cost too. The set up of the distributed storage system should also allow for securely perform computations on the shared data.

Contributions. In this paper, we address the above problems by introducing a decision support protocol run by the TA. The protocol guides data owners in choosing high-performing storage servers, and in setting up the most cost-efficient distributed storage system configuration that also meets the desired confidentiality and availability requirements. The suggested configuration also suggests the best-suited secret sharing scheme, and takes into account the hierarchical structure of the organization deploying it. Our decision support protocol assumes the existence of a performance scoring mechanism for selecting the highest-performing storage servers. Such a performance scoring mechanism has been introduced in previous work [34]. In terms of the underlying proactive secret sharing schemes, both non-hierarchical and hierarchical scenarios are supported. We include:

- The (non-hierarchical) proactive Herzberg version [12] of Shamir's secret sharing scheme [30];
- The proactive versions [35] of Tassa's two hierarchical secret sharing schemes [33].

We selected these specific secret sharing schemes for performance reasons. They are polynomial-based, which leads to fast reconstruction, unlike other prominent schemes such as the ones by Brickell [5] and Simmons [31]. However, the decision support protocol itself is not tied to any specific secret sharing scheme. In this sense, this part of our contribution can be seen as a generic framework. Concretely, to formalize the data owner requirements input to the decision support protocol, we model confidentiality, availability and cost constraints using linear programming. To evaluate the practicability of using constraint solving in this setting, we perform experiments on sixty data centers from six major cloud providers. Our results demonstrate that configurations optimizing the given constraints are found within minutes at most. The second part of our evaluation focusses on the time overhead due to the use of hierarchical secret schemes. Such schemes can usefully mirror the hierarchical structure of an organization, but this feature incurs additional complexity. To this end, we compare the run times of individual algorithms of the three above secret sharing schemes. We find that hierarchical secret sharing schemes cause only minimal performance loss for the online phase. For the offline phase, we observe a more significant slowdown, but this does not affect data owners directly.

Outline. The rest of the paper is organized as follows. We start by discussing related work with respect to hierarchical secret sharing (HSS) and outsourced multi-party computation (Sec. 2). Next, we recall preliminary notions about distributed storage systems based on multiple SSPs, Shamir (extended to a proactive scheme by Herzberg) secret sharing and Tassa (extended by Traverso) conjunctive and disjunctive secret sharing (Sec. 3). We then describe our core contribution: a decision support system guiding data owners in the selection of a suitable secret sharing scheme, in the selection of storage servers and in the allocation of shares between the different SSPs and storage servers (Sec. 4). Next, we evaluate our approach by implementing the integer linear programming problem introduced in the previous section using PuLP, and applying the GLPK linear problem solver to it (Sec. 5). Based on the evaluation's result, a discussion comparing candidate secret sharing schemes follows (Sec. 6). We then conclude (Sec. 7).

2 RELATED WORK

We now survey related work about HSS and cloud multi-party computation (MPC).

Hierarchical Secret Sharing. HSS [10] addresses scenarios where the shareholders are not equal in their reconstruction ability with respect to the shared message. For instance, in companies employees are organized in hierarchical levels and the shares they receive should reflect this structure. The first solution for HSS was proposed by Shamir [30], where the higher the level a shareholder is assigned to the more shares it gets. This approach overloads the most powerful shareholders, both from the storage space consumption and the safeguard of the shares themselves. Instead, approaches by Brickell [5] and Simmons [31] manage to distribute to each shareholder one share only. Generated shares differ with respect to how informative they are. More informative shares are distributed to shareholders assigned to higher levels, and vice versa. However, these two solutions are inefficient and, thus, they cannot be used in practice. In [33], Tassa proposed polynomial-based HSS schemes that generate shares of the same lengths and are comparable to Shamir's secret sharing scheme in terms of efficiency. Furthermore, Traverso et al. in [35] and [36] provided algorithms for Tassa's secret sharing schemes for renewing the shares, changing the access rules, and performing operations on shared data. These algorithms equip Tassa's scheme with the same functionalities that Shamir's secret sharing provides and that are valuable when it comes to outsourcing the storage of data.

Outsourcing Multi-Party Computation. In [17], the problem of outsourcing MPC to the cloud computing that carries out computations as a service was first considered. In particular, secure MPC is formalized in a server-aided setting, where an untrusted server performs computations for computationally weak users without learning any input nor the output through garbled circuits. A practical perspective of this problem is provided by the same authors in [18] and they discuss computations outsourced to multiple physical machines in [19]. In [20], it is discussed how the security assumptions of outsourced MPC and the multiplication property of the underlying secret sharing scheme can be weakened when trusted hardware and a semi-trusted third party are provided and the computations are performed my multiple cloud providers rather than one. Instead of using secret sharing, in [25] a solution based on additively homomorphic encryption where every user has its own pair of private-public key is presented. The server where these encrypted data are outsourced performs the requested computation without breaking confidentiality. In [16], a solution for outsourcing MPC to multiple untrusted servers relying on MACs is proposed, which ensures confidentiality when at least one server is trusted. Our approach differs from the works discussed above: we address the problem of outsourcing MPC on multiple physical machines where no encryption nor key management are involved and the underlying secret sharing scheme is not weakened in its properties nor augmented with additional schemes. Thus, the majority of the parties has to be honest to ensure security. Instead, scores are computed from performance and availability ratings, which are gauged by a trusted third party, the TA.

3 PRELIMINARIES

In this section, preliminary notions of distributed storage systems based on multiple SSPs are provided in Sec. 3.1 and the secret sharing primitives enabling such outsourced storage are presented in Sec. 3.2.

3.1 Distributed Storage Systems

Data storage is one of the main services offered in cloud computing to users. When document owners do not have the resources to store large amounts of data, they outsource them to multiple storage servers owned by different commercial SSPs by leveraging the secret sharing primitive to achieve confidentiality and availability. Secret sharing-based distributed storage systems [21, 32] are a solution for long-term data protection that do not involve encryption prior to data outsourcing. Thus, they do not introduce additional challenges with respect to key protection and key management nor weaknesses with respect to unbounded attackers. The future existence of an enhanced storage service built upon multiple cooperating providers is motivated by two main arguments. First, it is at present affordable to have multi-CSPs applications and services. Infrastructure as Code [39] makes it possible to programmatically define multi-provider resource allocations and application modules. This simplifies not only the development, but also operations on different CSPs. In particular, solutions to build [6, 11, 23] and configure [26, 27] cloud services across multiple CSPs already exist. Second, CSPs strongly differentiate their offers to justify their presence on the market [9]. Spot instances, serverless computing, and reserved instances [37] provide Infrastructure as a Service (IaaS) resources at different trade-offs in terms of performance, latency, and pricing. For example, cheap and revocable spot instances offer the possibility to obtain computational power at a lower price. Thus, SSPs hosted on data center from different CSPs may offer different solutions for distributed storage. For simplicity, in this paper we consider the model where each SSP owns its own data centers and offer to data owners several storage servers where to store the shares.

3.2 Suitable Secret Sharing Schemes for Distributed Storage Systems

Due to space constraints, we only recall the seminal, non-proactive schemes by Shamir and Tassa here. Their proactive versions are described in Appendix B. *Secret sharing* is a cryptographic primitive that generates shares of a message such that specific subsets of those shares are necessary to reconstruct the message. Shamir's and Tassa's secret sharing schemes (with their proactive versions) are suitable for the distributed storage systems described in Sec. 3.1.

3.2.1 Shamir's Scheme. Secret sharing-based distributed storage systems are generally based on this perfectly private threshold secret sharing scheme proposed by Shamir [30]. Let n be the number of storage servers $S_1, S_2, \ldots, S_n$ to which the shares are distributed, $t \leq n$ be the threshold required for reconstruction, and $\mathbb{F}_q$ be a field with $q > n$ elements. Shamir's secret sharing scheme relies on the fact that, in a field, a polynomial of degree $t-1$ is determined uniquely by at least t points on it. However, knowing only $t-1$ or less points, one cannot reconstruct the polynomial. Secret sharing is carried out with two protocols: Share, which takes a message m as input and produces n shares, and Reconstruct, which reconstructs the original message m from any subset of t shares as input. To share a message $m \in \mathbb{F}_q$, the data owner chooses a polynomial

$f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$ of degree $t-1$, such that $f(0) = m$ and $a_1, \ldots, a_{t-1}$ are chosen uniformly at random in $\mathbb{F}_q$. Typically, the data owner computes and distributes $\sigma_i := f(i)$ to storage servers S_i for $i = 1, \ldots, n$, where σ_i is referred to as a *share*. In the following, we denote this sharing algorithm as Share.

3.2.2 Tassa's HSS Schemes. The *conjunctive* and *disjunctive* HSS schemes proposed by Tassa [33] are the first HSS schemes based on Birkhoff interpolation of polynomials. Shares are either points on a polynomial or points on one of the derivatives of such polynomial. A hierarchy is composed of levels $L_0, \ldots, L_\ell$, where L_0 is the highest level, L_ℓ the lowest, and $\ell \leq n$. The cardinality of each level L_h is denoted by n_h and each shareholder is assigned to one level only. In addition, a threshold t_h is associated with each level L_h, for $h \in 0, \ldots, \ell$, such that $0 < t_0 < \cdots < t_\ell$. Tassa individuated two types of access structures, defining, respectively the conjunctive and the disjunctive HSS. The conjunctive access structure determines that a subset $A \subset S$ is authorized if, *for all levels* L_h, it contains t_h shareholders assigned to levels equal or higher than L_h, for $h = 0, \ldots, \ell$. The disjunctive access structure specifies that a subset $A \subset S$ is authorized if, *for at least one level* L_h, it contains t_h shareholders assigned to levels equal or higher than L_h, for $h = 0, \ldots, \ell$. In the following, we write information relating to disjunctive HSS in brackets. For conjunctive (disjunctive) HSS schemes the unique ID of shareholder $s_{i,j} \in S$ is a pair $(i, j) \in I \times I$, where $i = 1, \ldots, n_h$ and $j := t_{h-1}$ ($j := t_\ell - t_h$), for $h = 0, \ldots, \ell$ with $t_{-1} := 0$ and $t := t_\ell$. The algorithms Share and Reconstruct of the conjunctive (disjunctive) HSS are as follows.

Share takes as input a message $m \in \mathbb{F}_q$ and generates a polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$ of degree $t-1$, where $a_0 := m$ ($a_{t-1} := m$) and the coefficients $a_1, \ldots, a_{t-1} \in \mathbb{F}_q$ ($a_0, \ldots, a_{t-2} \in \mathbb{F}_q$) are chosen uniformly at random. It outputs share $\sigma_{i,j} \in \mathbb{F}_q$ for shareholder $s_{i,j} \in S$ computed as $\sigma_{i,j} := f^j(i)$, where $f^j(x)$ is the j-th derivative of polynomial $f(x)$ and pair $(i, j) \in I \times I$ is the unique ID of shareholder $s_{i,j} \in S$, for $i = 1, \ldots, n_h$ and $h = 0, \ldots, \ell$.

Reconstruct takes as input a set of shares held by a subset $R \subset S$ of shareholders. If R is unauthorized, i.e. $R \notin \Gamma$, then it outputs $\perp$. If R is authorized, i.e. $R \in \Gamma$, then it reconstructs polynomial $f(x)$ using Birkhoff interpolation and outputs $m = a_0$ ($m = a_{t-1}$).

The Birkhoff interpolation problem is a generalization of the Lagrange interpolation problem and describes the problem of finding a polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$ satisfying the equalities $f^j(i) = \sigma_{i,j}$. Given an authorized set $R \in \Gamma$ of shareholders for conjunctive (disjunctive) HSS schemes, the Birkhoff interpolation problem can be solved as follows. The *interpolation matrix* associated to set R is a binary matrix E where entry $e_{i,j}$ is 1 if shareholder $s_{i,j}$ participates with share $\sigma_{i,j}$ and 0 otherwise. We denote by $I(E) = \{(i, j) \text{ such that } e_{i,j} = 1\}$ the set containing the entries of E in lexicographic order, i.e. the pair (i, j) precedes the pair (i', j') if and only if $i < i'$ or $i = i'$ and $j < j'$. The elements of $I(E)$ are denoted by $(i_1, j_1), (i_2, j_2), \ldots, (i_r, j_r)$, where $r := |R|$. Furthermore, we set $\varphi := \{\phi_0, \phi_1, \phi_2, \ldots, \phi_{t-1}\} = \{1, x, x^2, \ldots, x^{t-1}\}$ and denote by ϕ_k^j the j-the derivative of ϕ_k, for $k = 0, \ldots, t-1$. Then the matrix $A(E, X, \varphi)$ is defined as follows:

$$A(E, X, \varphi) = \begin{pmatrix} \phi_0^{j_1}(i_1) & \phi_1^{j_1}(i_1) & \phi_2^{j_1}(i_1) & \cdots & \phi_{t-1}^{j_1}(i_1) \\ \phi_0^{j_2}(i_2) & \phi_1^{j_2}(i_2) & \phi_2^{j_2}(i_2) & \cdots & \phi_{t-1}^{j_2}(i_2) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \phi_0^{j_r}(i_r) & \phi_1^{j_r}(i_r) & \phi_2^{j_r}(i_r) & \cdots & \phi_{t-1}^{j_r}(i_r) \end{pmatrix}.$$

Then polynomial $f(x)$ can be reconstructed by computing

$$f(x) = \sum_{k=0}^{t-1} \frac{\det(A(E, X, \varphi_k))}{\det(A(E, X, \varphi))} x^k,$$

where $A(E, X, \varphi_k)$ is obtained from $A(E, X, \varphi)$ by replacing its $(k+1)$-th column with the shares $\sigma_{i,j}$ in lexicographical order.

4 A DECISION SUPPORT SYSTEM FOR LONG-TERM STORAGE AND MPC

Due to a wealth of choices, it is difficult for data owners to make informed decisions regarding which storage servers from what SSPs to include in their distributed storage system, which secret sharing scheme to select, and share allocation to specific SSPs. We now introduce a decision support system, enabled by a TA, to guide data owners in establishing a distributed storage system. The provided guidance includes the configuration of the shares distribution to be adopted, given confidentiality, availability, and cost requirements of the data owner. It also includes the most appropriate underlying secret sharing scheme to be used, and its parameters.

We require the following two assumptions:

(1) The data owner has computational access to the storage servers, i.e. the data owner is not limited to exclusive storage usage, but can run computations on the storage servers. This is crucial for the distributed storage system to periodically run proactive secret sharing to refresh the shares, but also to enable the data owners to perform computations on shared data through MPC protocols.
(2) A performance scoring mechanism run by the TA already exists. This mechanism is run periodically among all the storage servers from all the SSPs available in the marketplace (not just the ones included in the distributed storage system) to compute up to date performance figures. We stress that here performance is meant in a broad sense and the performance figures reflect the overall quality of service of a provider, as NIST has suggested [14]. We previously introduced a peer-rating-based performance scoring mechanism where a TA is in charge of the accuracy of the outputs [34].

To offer data owners effective decision support, the TA acts both as a mediator and as a service orchestrator. On the one hand, the TA is the mediator of the performance scoring mechanism. It decides when to run the mechanism, and ensures it leads to scores reflecting the actual performance of the storage servers. On the other hand, the TA is the centralized point of orchestration of SSPs because it provides an ad hoc plan of which secret sharing scheme to adopt, which storage servers to select and where to place the shares.

In Sec. 4.1, the step-by-step protocol for the decision support system is introduced. In Sec. 4.2, the confidentiality, availability, and cost requirements are modeled.

4.1 Decision Support Protocol

In the following, we detail the decision support protocol enabled as a service by a TA to guide data owners in the choice of the storage servers making up a distributed storage system. Our protocol principals are the data owner, the TA, and the storage servers from multiple SSPs. The data owner wants to perform MPC over shared documents, and to this end they agree on confidentiality and availability constraints, and how the MPC kernel should be implemented (e.g. the MPC kernel of [4]). The TA enables the decision support system by running a performance scoring mechanism and provides a detailed executing plan for data owners that optimizes the cost of the SSPs. As an example, we assume that Herzberg/Shamir's secret sharing scheme, Tassa's conjunctive HSS scheme, and its disjunctive variant are all options that the TA considers as possible solutions for distributed storage. The choice of candidate secret sharing schemes does not affect the general process. The protocol among the clients, the TA, and the SSPs is as follows.

(1) A data owner decides the confidentiality and the availability requirements of the distributed storage system it wants to build. It is assumed that it wants to minimize the cost of the distributed storage system given such requirements.
(2) The data owner sends a request to the TA with these specifics.
(3) The TA takes into account these requirements and the performance figures of the storage servers in the marketplace. Here it is assumed that the performance figures are up to date, i.e. that the TA has recently run the performance scoring mechanism in place.
(4) The TA returns to the data owner the following criteria, which minimize cost given data owner requirements.
    - the secret sharing scheme to be used (i.e. Herzberg/Shamir secret sharing, conjunctive Tassa secret sharing, or its disjunctive variant) and list of the corresponding parameters (number of shares n, number of levels and respective thresholds, if hierarchical);
    - a list of high-performing storage servers (from different SSPs) to be deployed;
    - the amount and type of shares to be stored within each storage server (and, thus, SSP).
(5) The data owner follows the criteria and runs the sharing algorithm to store into the distributed storage system the data to be outsourced.
(6) When enough storage servers are available, the data owner can run the MPC protocol according to what is the secret sharing scheme used to build the storage system.

This protocol can be adapted to a multi-user scenario, with the caveat that the data owners willing to deploy the same distributed storage system must agree on confidentiality and availability constraints before sending the request to the TA. Furthermore, when running computations of shared data with an MPC protocol, each data owner gets from the storage servers the partial results and interpolates them locally to get the final result of the computation. The computational overhead for long-term storage and MPC depends on the reconstructing threshold and is independent on the number of data owners leveraging the distributed storage system.

4.2 Constraint Modeling

In this section, we formalize constraints modelling secret sharing scheme instantiation and shares allocation across the storage servers of multiple SSPs. We model the problem of secret sharing scheme identification through integer linear programming (ILP) [24]. We design a set of ILP programs where the TA finds the secret sharing scheme matching the most cost-efficient configuration, given the data owner's constraints in terms of confidentiality and availability of the outsourced data. We target secret sharing schemes that support MPC and shares renewal in polynomial time, which makes their execution feasible in a reasonable amount of time when the reconstructing threshold t is not too large. These are Herzberg/Shamir secret sharing scheme, Tassa's conjunctive HSS, and its disjunctive variant. In the following, we denote by $x_{i,j,k}$ the number of shares of level j distributed to storage server S_i of SSP k. Integer N is the number of storage servers from all M SSPs in the marketplace. For all the three sharing schemes that we take into account, the objective function to minimize is:

$$\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{k=1}^{M} x_{i,j,k}, \tag{1}$$

which is subjected to constraints in terms of the adversary model, confidentiality, reconstruction, fault tolerance, and provider diversity. In the following, we show how to instantiate these constraint for Herzberg/Shamir's secret sharing scheme, and for Tassa's conjunctive and disjunctive HSS schemes. For both of Tassa's schemes, we set the amount H of spanned hierarchical levels to 3 for two reasons. On the one hand, three levels are sufficient to understand how the above mentioned constraints are instantiated for the HSS schemes even if more levels were spanned. On the other hand, a larger amount of spanned hierarchical levels would imply a higher reconstructing threshold. This leads to both higher computational and communication overhead (quadratic in the reconstructing threshold), which would make the execution of MPC unpractical, as pointed out in [4].

4.2.1 Adversary Model Constraint. This constraint is typically assumed when considering practical applications of the secret sharing primitive and MPC [3]. Normally, the adversary model taken into account is honest but curious. This means it seeks information about the outsourced data by corrupting the storage servers. However, it does not have enough power to make the storage servers deviate from the protocols they are supposed to run. In order to cope with a honest but curious type of adversary, an honest majority is assumed. This assumption entails a lower bound on the total amount n of shares generated. Given the reconstructing threshold t, the bound for all three considered secret sharing schemes is $n \geq 2t - 1$.

4.2.2 Confidentiality Constraints. They are set to ensure that none of the SSPs across their storage servers included in the distributed storage system have enough shares to successfully to reconstruct the outsourced data by themselves. We denote by $C_{1,k}, C_{2,k}, C_{3,k}$ the number of shares of, respectively, level L_1, L_2, L_3 distributed across all storage servers of one SSP, i.e. $C_{j,k} = \sum_{i=1}^{N} x_{i,j,k}$, for $j \in \{1, 2, 3\}$ and $k = 1, \ldots, M$.

For Herzberg/Shamir and disjunctive Tassa, there is one confidentiality constraint for each level. Recall from Sec. 3.2.2 that,

while the reconstructing threshold t corresponds to threshold t_1 of the highest level L_1, all levels have enough information to independently reconstruct the document, even though more and more shares are requested. Thus, the confidentiality constraints for Herzberg/Shamir and Tassa's schemes are:

$$C_{j,k} \leq t_j - 1 \quad j \in \{1, 2, 3\} \tag{2}$$

For Tassa's conjunctive HSS, there are $H - 1$ confidentiality constraints because the value of C_1 is bounded by threshold t_H of the lowest level L_H, which corresponds to reconstructing threshold t. Recall that thresholds t_{H-j} determine how many shares among the t_H shares must be from levels $L_1, \ldots, L_{H-j}$, for $j = 1, \ldots, H-1$. The two confidentiality constraints are:

$$\begin{cases} C_{2,k} \leq t_2 - C_{1,k} - 1 \\ C_{1,k} + C_{2,k} + C_{3,k} \leq t_3 - 1 \end{cases} \tag{3}$$

Similar constraints (2) and (3) have to be set for each SSP k, i.e. for $k = 1, \ldots, M$.

Summing up, Tassa's disjunctive HSS scheme requires $H \cdot M$ constraints, while its conjunctive variant requires $(H-1) \cdot M$ constraints.

4.2.3 Reconstruction Constraints. They are set to ensure that the shares distributed across the storage servers carry enough information to reconstruct the document. That is, they ensure that enough shares from each level are generated. We denote by R_1, R_2, R_3 the number of shares of, respectively, level L_1, L_2, L_3 distributed across all storage servers of all SSPs, i.e. $R_j = \sum_{i=1}^{N} \sum_{k=1}^{M} x_{i,j,k}$, for $j \in \{1, 2, 3\}$. For Herzberg/Shamir and disjunctive Tassa, there is one reconstruction constraint for each level. In the case of three levels, the three reconstruction constraints are:

$$R_j \geq t_j \quad j \in \{1, 2, 3\} \tag{4}$$

For conjunctive Tassa, there is one reconstruction constraint for each level. In the case of three levels, the three reconstruction constraints are:

$$\begin{cases} R_1 \geq t_1 \\ R_2 \geq t_2 - R_1 \\ R_3 \geq t_3 - (R_1 + R_2) \end{cases} \tag{5}$$

4.2.4 Fault Tolerance Constraints. They are set to reach a higher level of availability and ensure that the document is reconstructed even when a certain amount, say f, of SSPs is not available. We denote by F_1^f, F_2^f, F_3^f the number of shares of, respectively, level L_1, L_2, L_3 distributed across all endpoints (i.e., storage servers) of all M SSPs except f of them, denoted by $k_1, \ldots, k_f$, that is:

$$F_j^f = \binom{M}{f} \sum_{i=1}^{N} \sum_{k=1, k \notin \{k_1, \ldots, k_f\}}^{M} x_{i,j,k}, \quad j \in \{1, 2, 3\}. \tag{6}$$

Then, the fault tolerance constraints are the same as the reconstruction constraints where F_1^f, F_2^f, F_3^f are used instead of R_1, R_2, R_3 in (4) and (5). The fault tolerance constraints are a generalization of the reconstruction constraint with $f = 0$.

4.2.5 Provider Diversity Cost Constraint. From a data owner's perspective, the optimal solution for (1) is the one minimizing the number of SSPs. The reason is that transmission of data between storage servers of different providers is needed to perform MPC and the renewal of shares. However, this entails costs for the data owner, while communication between storage servers from the same provider is free. It is in the user's interest to minimize this cost by storing as many shares as possible (up to the reconstructing threshold t) in one SSP. G_k denotes a Boolean variable representing the fixed transmission cost payed by the user, which is 1 when at least one endpoint of SSP k is chosen and is 0 otherwise. We denote by N_k the amount of shares of all levels distributed across the endpoints of a SSP k. Then, for each SSP k, the diversity cost constraints are:

$$\frac{1}{(t-1)N_k} \sum_{i=1}^{N} \sum_{j=1}^{H} x_{i,j,k} \leq G_k \quad k = 1, \ldots, M. \tag{7}$$

The constraints for the transmission cost among endpoints of different SSPs are modeled similarly to what happen when communicating among different cloud service providers.

5 EVALUATING THE FEASIBILITY OF THE CONSTRAINT SATISFACTION APPROACH

We tested the protocol presented in Sec. 4 to investigate: (i) the running time to output the criteria for the most cost-efficient distributed storage system with different types of market configurations, and (ii) which secret sharing scheme leads, on average, to such solution. We implemented the ILP problem modeled as described in Sec. 4.2 in Python 3 by using the PuLP package [22]. PuLP provides an interface to define linear programs, which are decoupled from the underlying LP-Solver. We used the linear problem solver GLPK to evaluate performance of our implementation. Evaluations were run on an Intel i7-3612QM CPU at 2.10GHz, with 8 GB of RAM. We involved sixty data centers from six of the major public cloud providers (AWS, OVH, Google, IBM, Microsoft, and Deutsche Telekom) and evaluated our protocol with different distributions of endpoints (i.e. storage servers in our model) across the SSPs. For instance, we tested configurations where there are few SSPs in the market owning many storage servers (e.g. 12 SSPs with 10 storage servers each) and configurations where there are many SSPs in the market owning few storage servers (e.g. 90 SSPs with 2 storage servers each). The data owner gives as input the desired confidentiality and fault tolerance constraints defined in Sec. 4.2, where for our tests the reconstructing threshold t is between 2 and 10 and the amount f of possible unavailable SSPs is between 0 and 2. The reason behind these choices is that, on the one hand, a reconstructing threshold t of 10 is high enough to see how the behavior of the optimal solution evolves and higher values of t would make the system impractical due to the communication overhead generated during computations [4]. On the other hand, the availability of current CSPs suggests that, in a distributed system, it is unlikely that more than two SSPs hosted on cloud providers are unavailable simultaneously.

The results are presented in Fig. 1, which shows how the time necessary to output the optimal solution for the instantiation of a distribute storage system is larger for higher fault tolerance. This

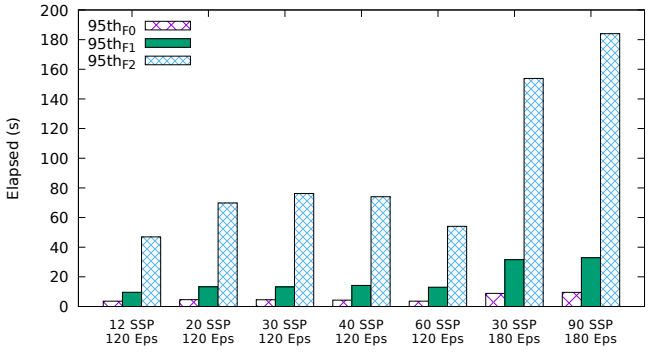


Figure 1: Running times to determine the most cost-efficient solutions for different types of market configurations and fault tolerance requirements. In 95% of the test runs, such a solution can be found at most in seconds for no fault tolerance or fault tolerance 1 for any type of market configuration. Finding a solution with fault tolerance 2 requires at most a few minutes. "Eps" stands for "Endpoints".

is in particular remarkable for solutions with fault tolerance 2. The amount of constraints to fulfill increases drastically when the fault tolerance is greater than 1 because of the binomial coefficient involved in the constraints. However, if the data owner requires no fault tolerance or fault tolerance 1, which already leads to high availability, the most cost-efficient solutions is output in a matter of seconds. This is an affordable amount of time that users can wait for in a realistic scenario.

6 EVALUATING RUN TIMES OF CANDIDATE SECRET SHARING SCHEMES

In this section, we provide the first comparison in terms of run time between Herzberg/Shamir's secret sharing scheme and (the proactive versions of) both of Tassa's HSS schemes. In Sec. 3, we discussed how the HSS schemes proposed by Tassa [33] lead to distributed storage systems addressing scenarios where the shareholders are organized in a hierarchical structure (such as companies, hospitals, and institutions). The main reasons why Tassa's schemes are good building blocks for distributed storage systems are the following. On the one hand, all the shares generated have the same length of the data, meaning that more informative shares are not longer than less informative shares. This way, all storage servers making up the distributed storage system allocate the same storage space for the outsourcing of data. On the other hand, Tassa's HSS schemes are polynomial-based primitives whose Reconstruction (see Sec. 3.2) algorithm relies on solving a Birkhoff interpolation problem. This is a generalization of the Lagrange interpolation problem used to reconstruct the secret message in Herzberg/Shamir's secret sharing scheme. We demonstrate in this section that the run times for Tassa's HSS is comparable to Shamir's secret sharing scheme, as shown in Fig. 2.

The simulations where performed for (t, n) values $(2, 3), (3, 5), (4, 7), (5, 9)$, and $(6, 11)$. With respect to Tassa's conjunctive and disjunctive secret sharing schemes, for every pair (t, n), all possible hierarchical configurations with up to t levels were tested and the average of the run time is what is displayed in the corresponding plots of Fig. 2. The files to be shared were of size 100 Byte, 1 KB, and 10 KB. A 256 byte encoding was tested, where every 256 characters from the file were converted to an integer in the finite field $\mathbf{F}_{2^{2203}-1}$. Text files ranges from 100 byte to 100 kilobyte were used for the evaluations. Measurements are taken in seconds for the execution of each algorithm alone by using the internal clock time of the processor. The simulations were performed on an Intel (Quad-Core) i5-8250U CPU clocked at 1.6GHz, with 8 GB of RAM.

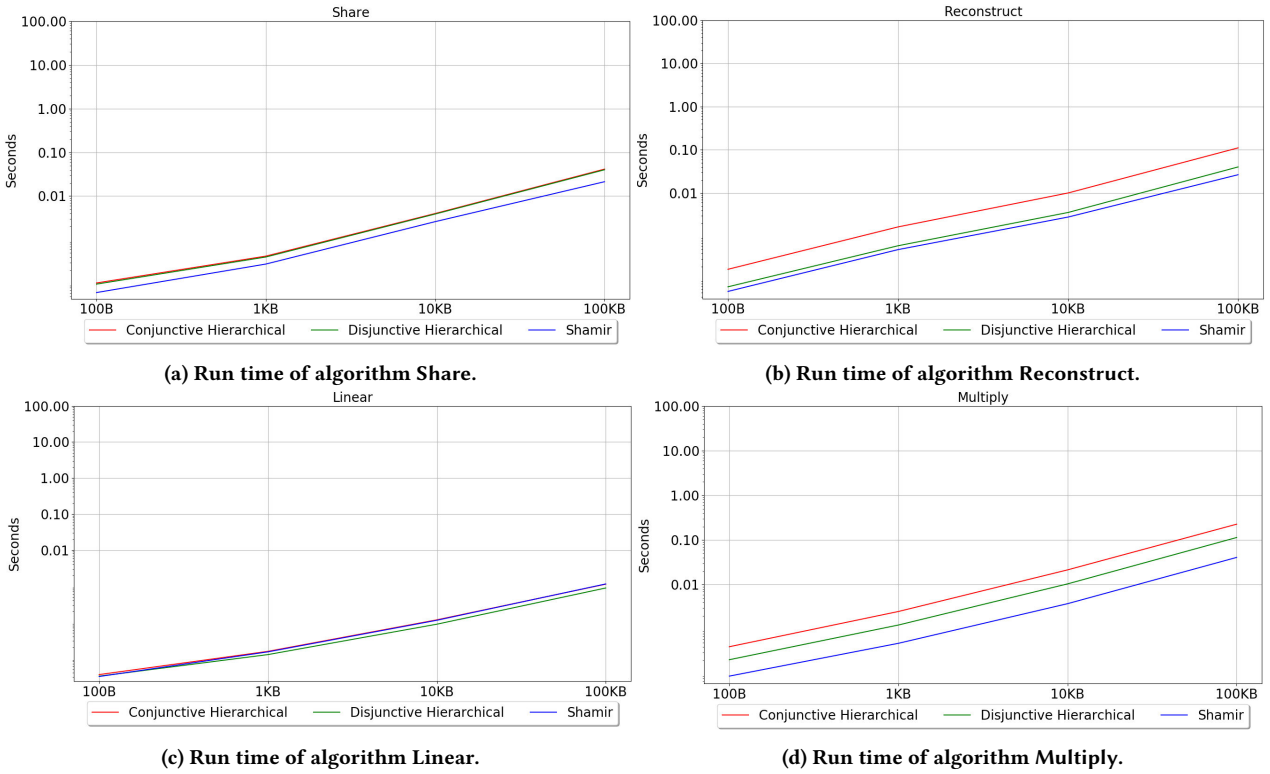
In Fig. 2, we compare the run times of algorithms Share for the Herzberg/Shamir, conjunctive Tassa and disjunctive Tassa schemes. For Shamir's scheme, that means to compute $t-1$ multiplication for each share to be generated for a polynomial of degree $t-1$. Besides polynomials' evaluation, algorithm Share of Tassa's HSS schemes requires also to compute up to $t-1$ polynomials' derivatives. However, the additional multiplications due to derivation are balanced by the fewer multiplications needed when evaluating derivatives of polynomials and this results into a slightly longer run time.

Algorithms Reconstruct of both Herzberg/Shamir and Tassa's schemes require Gaussian elimination to solve a system of t linear equations in order to reconstruct the polynomial used to share the message. Then, the message is retrieved through polynomial evaluation. However, in the secret sharing framework the only coefficient that matters is the free coefficient for Herzberg/Shamir and conjunctive Tassa and the last coefficient for disjunctive Tassa. For the HSS schemes, this implies that only determinants $\det(A(E, X, \varphi_0))$ $(\det(A(E, X, \varphi_{t-1})))$ and $\det(A(E, X, \varphi))$ have to be computed, where the latter can be computed in advance off-line. This leads to a complexity of $O(t^3)$ for matrix $A(E, X, \varphi)$ of dimension $t \times t$ in case the LU decomposition technique is used [1]. Also, Tassa's disjunctive's Reconstruct algorithm is faster than the Tassa's conjunctive counterpart because the matrix $A(E, X, \varphi_{t-1})$ has more zeros than $A(E, X, \varphi_0)$, leading to a faster computation of the determinants. The Reconstruction algorithm of Herzberg/Shamir's secret sharing requires $t-1$ multiplications. Fig. 2 shows that the three run times are comparable, especially those of Herzberg/Shamir's and Tassa's conjunctive secret sharing schemes.

Due to the relevance of MPC in this paper, in Fig. 2 we also provide run times of the algorithm to perform linear operations on (hierarchically) shared messages, referred to as Linear, and of the algorithm to perform multiplications on (hierarchically) shared messages, referred to as Multiply. In the following, we describe algorithms Linear and Multiply for Tassa's conjunctive and disjunctive HSS schemes, which were first provided by Traverso et al. [36]. These algorithms mirror the corresponding algorithms Linear and Multiply for Herzberg/Shamir [2], which can be easily derived by substituting below share $\sigma_{i,j}$ of a shareholder $s_{i,j}$ with share σ_i of shareholder s_i since all shares are equivalent and the shareholders belong to the same level.

Linear takes as input shares $\sigma_{i,j}(m_1), \sigma_{i,j}(m_2) \in \mathbb{F}_q$ held by shareholder $s_{i,j} \in S$, and scalars $\lambda_1, \lambda_2 \in \mathbb{F}_q$. It outputs share $\sigma_{i,j}(m) := \lambda_1 \cdot \sigma_{i,j}(m_1) + \lambda_2 \cdot \sigma_{i,j}(m_2) \in \mathbb{F}_q$ for shareholder $s_{i,j} \in S$.

Multiply takes as input shares $\sigma_{i,j}(m_1), \sigma_{i,j}(m_2) \in \mathbb{F}_q$ generated during algorithm Share and shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta), \sigma_{i,j}(\gamma) \in \mathbb{F}_q$ held by shareholder $s_{i,j} \in S$ generated during algorithm PreMult (see Appendix A). It outputs share $\sigma_{i,j}(m) \in \mathbb{F}_q$ for message $m = m_1 \cdot m_2$, which is computed performing the following steps.



(a) Run time of algorithm Share.

(b) Run time of algorithm Reconstruct.

(c) Run time of algorithm Linear.

(d) Run time of algorithm Multiply.

Figure 2: Run time of Share, Reconstruct, Linear, and Multiply for Herzberg/Shamir, conjunctive Tassa and disjunctive Tassa.

(1) Shareholder $s_{i,j}$ computes share $\sigma_{i,j}(\delta) := \sigma_{i,j}(m_1) - \sigma_{i,j}(\alpha)$ and share $\sigma_{i,j}(\varepsilon) := \sigma_{i,j}(m_2) - \sigma_{i,j}(\beta)$ using algorithm Linear.
(2) Shareholders from an authorized set $R \in \Gamma$ run algorithm Reconstruct with shares $\sigma_{i,j}(\delta)$, $\sigma_{i,j}(\varepsilon)$ as input to publicly reconstruct values δ, ε using the bulletin board.
(3) Shareholder $s_{i,j} \in S$ computes the share $\sigma_{i,j}(m) := \sigma_{i,j}(\gamma) + \varepsilon \cdot \sigma_{i,j}(m_1) + \delta \cdot \sigma_{i,j}(m_2) - \delta\varepsilon$ using algorithm Linear.

The run times of Linear for Herzberg/Shamir's secret sharing scheme and for Tassa's conjunctive and disjunctive HSS scheme are practically the same. The run time for algorithm Multiply is comparable as well. The gap between the run times of algorithm Multiply for Herzberg/Shamir and for Tassa's schemes is due to the gap in run time between the corresponding algorithms Reconstruct. Algorithm Multiply does not involve multiplications, but only linear operations. That is because algorithm Multiply describes the steps to be performed during the on-line phase only, i.e. once certain multiplications are requested from the data owner. The most intensive operations are instead carried out during the off-line phase by RandShares and PreMult (see Appendix A). This allows Tassa's conjunctive and disjunctive HSS schemes to be as efficient as Herzberg/Shamir for operations over shared messages.

7 CONCLUSION

Proactive secret sharing schemes are expensive to deploy, but enable information-theoretic confidentiality in distributed storage systems protecting sensitive data sets. Data owners require guidance in selecting suitable participating servers and the best-suited secret sharing scheme. In this paper, we make the case for decision support systems providing guidance for secret sharing scheme selection and share allocation to storage service providers. We specify the decision support system as a protocol processing a set of confidentiality and availability constraints, input to a solver for integer linear programming problems. The approach was evaluated on two levels. First, we evaluated the practicality of the constraint solving problem on data centers from major public cloud providers. This first evaluation was agnostic with respect to the underlying secret sharing scheme. This first evaluation showed that an infrastructure configuration of storage servers fulfilling the specified requirements can be found in minutes at most. Second, we compared the run times of our candidate proactive secret sharing schemes, which are based on initial schemes by Shamir and Tassa. This second evaluation was performed independently of a specific cloud configuration. We discussed trade-offs between the performances of the considered secret sharing schemes for our specific setting. Our evaluation showed that the additional complexity stemming from the support of hierarchical structures in HSS only causes a minimal performance impact for the online phase. In some cases, even a performance gain is observed. The performance loss for the offline phase is more significant, but unproblematic in practice since it does not directly impact data owners. The gist of our approach is general, and it can be instantiated with both different constraints and other secret sharing scheme than the ones considered here.

REFERENCES

[1] Monika Agarwal and Rajesh Mehra. 2014. Review of Matrix Decomposition Techniques for Signal Processing Applications. *International Journal of Engineering Research and Applications* 4, 1 (2014), 90–93.

[2] Donald Beaver. 1991. Efficient Multiparty Protocols Using Circuit Randomization. In *CRYPTO (LNCS)*, Vol. 576. Springer, 420–432.

[3] Dan Bogdanov, Marko Jõemets, Sander Siim, and Meril Vaht. 2015. How the Estonian Tax and Customs Board Evaluated a Tax Fraud Detection System Based on Secure Multi-party Computation. In *FC 2015 (LNCS)*, Vol. 8975. Springer, 227–234.

[4] Dan Bogdanov, Sven Laur, and Jan Willemson. 2008. Sharemind: A Framework for Fast Privacy-Preserving Computations. In *ESORICS 2008 (LNCS)*, Vol. 5283. Springer, 192–206.

[5] Ernest F. Brickell and Douglas R. Stinson. 1990. Some Improved Bounds on the Information Rate of Perfect Secret Sharing Schemes. In *CRYPTO (LNCS)*, Vol. 537. Springer, 242–252.

[6] Cloudify. 2017. Cloud & NFV Orchestration Based on TOSCA. http://cloudify.co.

[7] D-Wave. 2015. Announcing the D-Wave 2x Quantum Computer. https://www.dwavesys.com/blog/2015/08/announcing-d-wave-2x-quantum-computer/.

[8] Ivan Damgård and Jesper Buus Nielsen. 2007. Scalable and Unconditionally Secure Multiparty Computation. In *CRYPTO (LNCS)*, Vol. 4622. Springer, 572–590.

[9] Yehia Elkhatib. 2016. Mapping Cross-Cloud Systems: Challenges and Opportunities. In *HotCloud 2016*. USENIX Association.

[10] Oriol Farràs and Carles Padró. 2010. Ideal Hierarchical Secret Sharing Schemes. In *TCC 2010 (LNCS)*, Vol. 5978. Springer, 219–236.

[11] HashiCorp. 2017. Terraform. http://terraform.io.

[12] Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. 1995. Proactive secret sharing or: How to cope with perpetual leakage. In *CRYPTO'95*. Springer, 339–352.

[13] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. 1995. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In *CRYPTO '95 (LNCS)*, Vol. 963. Springer, 339–352. https://doi.org/10.1007/3-540-44750-4_27

[14] Michael Hogan, Fang Liu, Annie Sokol, and Jin Tong. 2011. NIST Cloud Computing Standards Roadmap (NIST-SP 500-291). *NIST Special Publication* 35 (2011).

[15] Jingwei Huang and David M. Nicol. 2013. Trust mechanisms for cloud computing. *J. Cloud Computing* 2 (2013), 9.

[16] Thomas P Jakobsen, Jesper Buus Nielsen, and Claudio Orlandi. 2014. A Framework for Outsourcing of Secure Computation. In *ACM CCSW 2014*. ACM, 81–92.

[17] Seny Kamara, Payman Mohassel, and Mariana Raykova. 2011. Outsourcing Multi-Party Computation. Cryptology ePrint Archive, Report 2011/272. https://eprint.iacr.org/2011/272.

[18] Seny Kamara, Payman Mohassel, and Ben Riva. 2012. Salus: a system for server-aided secure function evaluation. In *CCS'12*. ACM, 797–808.

[19] Seny Kamara and Mariana Raykova. 2011. Secure outsourced computation in a multi-tenant cloud. 15–16.

[20] Jake Loftus and Nigel P. Smart. 2011. Secure Outsourced Computation. In *AFRICACRYPT 2011 (LNCS)*, Vol. 6737. Springer, 1–20.

[21] Thomas Loruenser, Andreas Happe, and Daniel Slamanig. 2015. ARCHISTAR: towards secure and robust cloud based data sharing. In *CloudCom 2015*. IEEE, 371–378.

[22] Stuart Anthony Mitchell. 2018. PuLP package and GLPK solver. https://pypi.python.org/pypi/PuLP.

[23] Alex Palesandro, Marc Lacoste, Nadia Bennani, Chirine Ghedira Guegan, and Denis Bourge. 2017. Mantus: Putting Aspects to Work for Flexible Multi-Cloud Deployment. In *CLOUD2017*. IEEE Computer Society, 656–663.

[24] Christos H Papadimitriou and Kenneth Steiglitz. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

[25] Andreas Peter, Erik Tews, and Stefan Katzenbeisser. 2013. Efficiently Outsourcing Multiparty Computation Under Multiple Keys. *IEEE Trans. Information Forensics and Security* 8, 12 (2013), 2046–2058.

[26] Puppet. 2019. http://www.puppet.com.

[27] Red Hat. 2019. Ansible. http://www.ansible.com.

[28] RSA. 2011. Press Release: RSA Establishes Cloud Trust Authority to Accelerate Cloud Adoption. https://www.emc.com/about/news/press/2011/20110214-01.htm.

[29] Faiza Samreen, Yehia Elkhatib, Matthew Rowe, and Gordon S. Blair. 2016. Daleel: Simplifying cloud instance selection using machine learning. In *NOMS 2016*. IEEE, 557–563.

[30] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (1979), 612–613.

[31] Gustavus J. Simmons. 1990. How to (Really) Share a Secret. In *CRYPTO' 88 (LNCS)*, Vol. 403. Springer, 390–448.

[32] Daniel Slamanig, Agi Karyda, and Thomas Lorünser. 2016. PRISMACLOUD — Privacy and Security Maintaining Services in the Cloud. *ERCIM News* 2016, 104 (2016), 46–46.

[33] Tamir Tassa. 2007. Hierarchical Threshold Secret Sharing. *J. Cryptology* 20, 2 (2007), 237–264.

[34] Giulia Traverso, Denis Butin, Johannes Buchmann, and Alex Palesandro. 2018. Coalition-Resistant Peer Rating for Long-Term Confidentiality. In *PST 2018*. IEEE Computer Society, 1–10.

[35] Giulia Traverso, Denise Demirel, and Johannes Buchmann. 2016. Dynamic and verifiable hierarchical secret sharing. In *ICITS 2016 (LNCS)*, Vol. 10015. Springer, 24–43.

[36] Giulia Traverso, Denise Demirel, and Johannes Buchmann. 2018. Performing Computations on Hierarchically Shared Secrets. In *AFRICACRYPT 2018 (LNCS)*, Vol. 10831. Springer, 141–161.

[37] Cheng Wang, Bhuvan Urgaonkar, Aayush Gupta, George Kesidis, and Qianlin Liang. 2017. Exploiting Spot and Burstable Instances for Improving the Cost-efficacy of In-Memory Caches on the Public Cloud. In *EuroSys 2017*. ACM, 620–634.

[38] Thomas Watson et al. 2018. A programmable two-qubit quantum processor in silicon. *Nature* 555, 7698 (2018), 633–637.

[39] Andreas Wittig and Michael Wittig. 2015. *Amazon Web Services in Action*. Manning Publications Co.

A RUN TIMES FOR MULTIPLICATION PREPROCESSING ALGORITHMS

The following algorithms RandShares and PreMult are adaptations of the algorithms for Shamir's secret sharing scheme proposed by Damgård and Nielsen [8] to fit the hierarchical setting. These are the off-line operations performed to compute the multiplication over shares secrets, i.e. to perform algorithm Multiply of Sec. 6.

During the off-line phase, a triple (α, β, γ) is generated such that the following conditions hold:

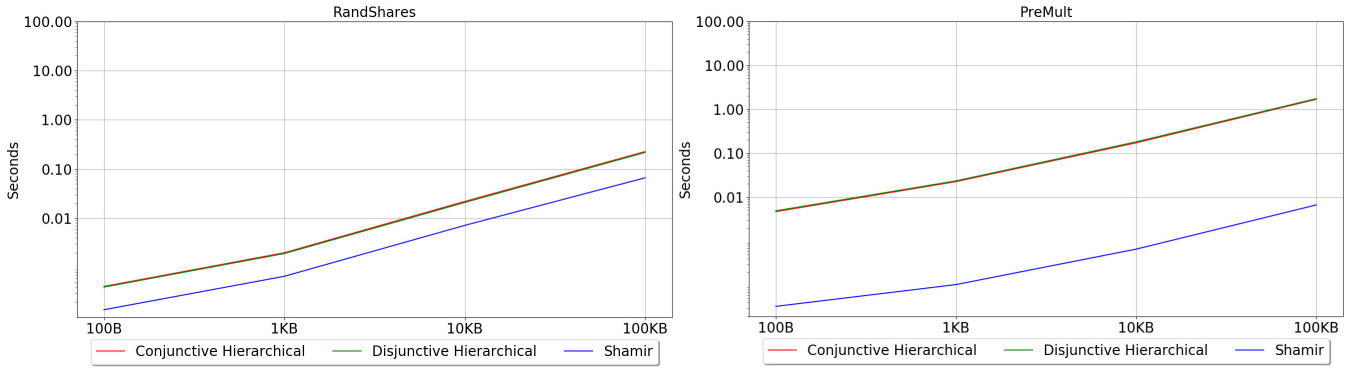
- $\alpha \cdot \beta = \gamma$.
- Each shareholder $s_{i,j} \in S$ with ID $(i,j) \in I \times I$ holds shares $\sigma_{i,j}(\alpha) := f_\alpha^j(i)$, $\sigma_{i,j}(\beta) := f_\beta^j(i)$, and $\sigma_{i,j}(\gamma) := f_\gamma^j(i)$, where $f_\alpha(x)$, $f_\beta(x)$, and $f_\gamma(x)$ are the polynomials of degree $t-1$ sharing α, β, and γ, respectively.

We present RandShares to compute shares $\sigma_{i,j}(\alpha)$ for α, but it can be run analogously to generate shares $\sigma_{i,j}(\beta)$ for β.

RandShares takes as input values $\alpha_{i,j} \in \mathbb{F}_q$ chosen uniformly at random by shareholders $s_{i,j} \in S$. It outputs shares $\sigma_{i,j}(\alpha)$ of message $\alpha \in \mathbb{F}_q$ for shareholders $s_{i,j} \in S$. To do that, each shareholder $s_{i,j} \in S$ has to perform the following steps.

(1) It chooses a secret message $\alpha_{i,j} \in \mathbb{F}_q$ uniformly at random.
(2) It runs Share to generate a polynomial $f_{\alpha_{i,j}}(x)$ of degree $t-1$ defined as $f_{\alpha_{i,j}}(x) := a_{0,(i,j)} + a_{1,(i,j)}x + \cdots + a_{t-1,(i,j)}x^{t-1}$, where $a_{0,(i,j)} = \alpha_{i,j}$ ($a_{t-1,(i,j)} = \alpha_{i,j}$) and $a_{1,(i,j)}, \ldots, a_{t-1,(i,j)} \in \mathbb{F}_q$ ($a_{0,(i,j)}, \ldots, a_{t-2,(i,j)} \in \mathbb{F}_q$) are chosen uniformly at random. Shares $\sigma_{i',j'}(\alpha_{i,j})$ for shareholders $s_{i',j'} \in S$ with ID $(i',j') \neq (i,j)$ are computed as $\sigma_{i',j'}(\alpha_{i,j}) := f_{\alpha_{i,j}}^{j'}(i')$. Share $\sigma_{i,j}(\alpha_{i,j})$ for shareholder $s_{i,j}$ itself is computed as $\sigma_{i,j}(\alpha_{i,j}) := f_{\alpha_{i,j}}^{j}(i)$.
(3) It sends shares $\sigma_{i',j'}(\alpha_{i,j})$ to shareholders $s_{i',j'} \in S$ with ID $(i',j') \neq (i,j)$ using a private channel and keeps share $\sigma_{i,j}(\alpha_{i,j})$.
(4) It runs Linear of Sec. 6 to compute share $\sigma_{i,j}(\alpha)$ using share $\sigma_{i,j}(\alpha_{i,j})$ and all the shares $\sigma_{i,j}(\alpha_{i',j'})$ received from shareholders $s_{i',j'}$ as $\sigma_{i,j}(\alpha) := \sum_{(i',j')\neq(i,j)} \sigma_{i,j}(\alpha_{i',j'}) + \sigma_{i,j}(\alpha_{i,j})$.

PreMult takes as input shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta)$ for each shareholder $s_{i,j} \in S$ computed by RandShares and outputs for each shareholder $s_{i,j} \in S$ a triple of shares $\sigma_{i,j}(\alpha), \sigma_{i,j}(\beta), \sigma_{i,j}(\gamma) \in \mathbb{F}_q$, such that for each triple it holds that $\sigma_{i,j}(\gamma) = \sigma_{i,j}(\alpha\beta)$. Each shareholder $s_l \in R$ from an authorized subset $R \in \Gamma$ performs the following steps:



(a) Run time of algorithm RandShares.

(b) Run time of algorithm PreMult.

Figure 3: Run times of RandShares and PreMult for Herzberg/Shamir, conjunctive Tassa and disjunctive Tassa.

(1) Use its shares $\sigma_l(\alpha)$ and $\sigma_l(\beta)$ and the unique ID (i,j) of shareholder $s_{i,j}$ to compute the values $\lambda^m_{l,(i,j)}$ and $\mu^m_{l,(i,j)}$ defined as:

$$\lambda^m_{l,(i,j)} := \sigma_l(\alpha) \sum_{k=m}^{j} \frac{k!}{(k-m)!}(-1)^{l-1+k} \frac{\det(A_{l-1,k}(E,X,\varphi))}{\det(A(E,X,\varphi))} i^{k-m},$$

and

$$\mu^m_{l,(i,j)} := \sigma_l(\beta) \sum_{k=m}^{j} \frac{k!}{(k-m)!}(-1)^{l-1+k} \frac{\det(A_{l-1,k}(E,X,\varphi))}{\det(A(E,X,\varphi))} i^{k-m},$$

where $m = 0, \ldots, j$ and $A(E,X,\varphi)$ and $A_{l-1,k}(E,X,\varphi)$ are the matrices defined in Sec. 3.

(2) Randomly split $\lambda^m_{l,(i,j)}$ and $\mu^m_{l,(i,j)}$ into r values, i.e. $\lambda^m_{l,(i,j)} = \lambda^m_{1,l,(i,j)} + \cdots + \lambda^m_{r,l,(i,j)}$ and $\mu^m_{l,(i,j)} = \mu^m_{1,l,(i,j)} + \cdots + \mu^m_{r,l,(i,j)}$.

(3) Send $\lambda^m_{u,l,(i,j)}$ and $\mu^m_{u,l,(i,j)}$ to shareholder $s_u \in R$, for $u = 1, \ldots, r$ and $u \neq l$, using a private channel.

(4) Collect all values $\lambda^m_{l,u,(i,j)}$ and $\mu^m_{l,u,(i,j)}$ received from shareholder $s_u \in R$, for $u = 1, \ldots, r$ and $u \neq l$, and computes $\delta^m_{l,(i,j)} := \sum_{u=1}^{r} \lambda^m_{l,u,(i,j)}$ and $\varepsilon^m_{l,(i,j)} := \sum_{u=1}^{r} \mu^m_{l,u,(i,j)}$, for $m = 0, \ldots, j$.

(5) Send $\delta^m_{l,(i,j)}$ and $\varepsilon^m_{l,(i,j)}$ to shareholder $s_{i,j}$ using a private channel.

Then, all shareholders within the set S compute their shares. Each shareholder $s_{i,j} \in S$ performs the following steps:

(1) Compute $\delta^m_{(i,j)} := \sum_{l=1}^{r} \delta^m_{l,(i,j)}$ and $\varepsilon^m_{(i,j)} := \sum_{l=1}^{r} \varepsilon^m_{l,(i,j)}$ using the values $\delta^m_{l,(i,j)}$ and $\varepsilon^m_{l,(i,j)}$, for $m = 0, \ldots, j$, received from shareholder $s_l \in R$, for $l = 1, \ldots, r$.

(2) Compute share $\sigma_{i,j}(\gamma)$ as

$$\sigma_{i,j}(\gamma) := \sigma_{i,j}(\alpha\beta) = \sum_{m=0}^{j} \binom{j}{m} \delta^{j-m}_{(i,j)} \cdot \varepsilon^m_{(i,j)}.$$

Fig. 3 shows run times for algorithms RandShares and PreMult both for Herzberg/Shamir and for conjunctive and disjunctive Tassa. For RandShares and PreMult, the run times for conjunctive and disjunctive Tassa are longer than for Herzberg/Shamir. However, these algorithms can be run in advance and do not affect the data owner directly.

B PROACTIVE SECRET SHARING SCHEMES

In the following, we present the proactive versions of Shamir's and Tassa's secret sharing schemes. We start with Herzberg's proactive version [13] of Shamir's scheme.

B.1 Proactive Herzberg/Shamir Secret Sharing

Share. Let $m \in \mathbb{F}_q$ be the message to be shared with shareholders $s_1, s_2, \ldots, s_n$, where $i \in \mathcal{I}$ is the unique ID of storage server S_i. After defining the polynomial $f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$, where $a_0 := m$ and $a_1, a_2, \ldots, a_{t-1} \in \mathbb{F}_q$ are chosen uniformly at random, the data owner computes share σ_i for shareholder s_i as $\sigma_i := f(i)$ and it sends it to shareholder s_i, for $i = 1, 2, \ldots, n$.

Renew. For each shareholder s_i performs the following steps, for $i = 1, \ldots, t$:

1. Select a polynomial $f_i(x) = a_{i,0} + a_{i,1}x + \cdots + a_{i,t-1}x^{t-1}$, where $a_{i,0} = 0$ and $a_{i,1}, a_{i,2}, \ldots, a_{i,t-1} \in \mathbb{F}_q$ are chosen uniformly at random.
2. Compute randomness value $\sigma_{j,i} := f_i(j)$ for shareholder s_j, for $j \neq i$, and randomness value $\sigma_{i,i} := f_i(i)$.
3. Send randomness value $\sigma_{j,i}$ to shareholder s_j for $j \neq i$, keep randomness value $\sigma_{i,i}$ private.
4. Receive randomness value $\sigma_{i,j}$ from shareholder s_j, for $j \neq i$.
5. Compute updated share σ'_i as $\sigma'_i := \sigma_i + \sum_{j=1}^{n} \sigma_{i,j}$.
6. Delete old share σ_i.

Reconstruct. t valid shares $\sigma_1, \sigma_2, \ldots, \sigma_t$ are used as input to reconstruct polynomial $f(x)$ by using Lagrange interpolation. The message is retrieved as $f(0) = m$.

B.2 Proactive Traverso/Tassa Secret Sharing

We now present Traverso's proactive version [35] of Tassa's conjunctive and disjunctive hierarchical secret sharing scheme. Everything is discussed for the conjunctive access structure; the equivalent for the disjunctive one can be found in brackets.

Share. Let $m \in \mathbb{F}_q$ be the message to be shared with shareholder $s_{i,j}$ with unique identity ID $(i,j) \in I \times I$. After defining polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1}x^{t-1}$, where $a_0 = m$ ($a_{t-1} = m$) and $a_1, \ldots, a_{t-1} \in \mathbb{F}_q$ ($a_0, \ldots, a_{t-2} \in \mathbb{F}_q$) are chosen uniformly at

random, the data owner computes share $\sigma_{i,j} = f^j(i)$ for shareholder $s_{i,j} \in L_h$, for $i = 1, \ldots, n_h$ and $h = 0, \ldots, \ell$.

Renew. Each shareholder $s_l \in R$ (where R is an authorized subset of shareholders of cardinality r) performs the following steps, for $l = 1, \ldots, r$:

(1) It computes its partial Birkhoff interpolation coefficient

$$a_{l,0} := \sigma_l (-1)^{l-1} \frac{\det(A_{l-1,0}(E, X, \varphi))}{\det(A(E, X, \varphi))}$$

$$\left(a_{l,t-1} = \sigma_l (-1)^{l+t-2} \frac{\det(A_{l-1,t-1}(E, X, \varphi))}{\det(A(E, X, \varphi))}\right).$$

(2) It chooses a polynomial $f_l(x) = a_{l,0} + a_{l,1}x + a_{l,2}x^2 + \cdots + a_{l,t-1}x^{t-1}$ of degree $t-1$, where $a_{l,0} = a_{l,0}$ ($a_{l,t-1} = a_{l,t-1}$) is the partial Birkhoff interpolation coefficient and $a_{l,1}, \ldots, a_{l,t-1} \in \mathbb{F}_q$ ($a_{l,0}, \ldots, a_{l,t-2} \in \mathbb{F}_q$) are chosen uniformly at random.

(3) It computes randomness $\sigma_{l,i,j}$ for shareholder $s_{i,j} \in S$ as $\sigma_{l,i,j} := f'^j_l(i)$.

(4) It sends randomness $\sigma_{l,i,j}$ to shareholder $s_{i,j} \in S$ using a private channel.

(5) It deletes its share.

Each new shareholder $s_{i,j} \in S$ computes its share $\sigma'_{i,j}$ adding all randomness $\sigma_{l,i,j}$ received, i.e. $\sigma'_{i,j} := \sigma_{i,j} + \sum_{l=1}^{r} \sigma_{l,i,j}$.

Reconstruct. It takes as input shares held by a subset $R \subset S$ of shareholders. If $R \in \Gamma$, it outputs m reconstructed using Birkhoff interpolation. It outputs $\perp$ otherwise.